

Working with Data in Python Cheat Sheet

Reading and writing files		
Package/Method	Description	Syntax and Code Example
File opening modes	Different modes to open files for specific operations.	Syntax: r (reading) w (writing) a (appending) + (updating) read/write b (binary, otherwise text) Examples: with open('data.txt', 'r') as file: content = file.read() print(content) with open('output.txt', 'w') as file: file.write('Hello, world!') with open('log.txt', 'a') as file: file.write('Log entry: Something happened.') with open('data.txt', 'r+') as file: content = file.read() file.write('Updated content: ' + content); file.close()
File reading methods	Different methods to read file content in various ways.	Syntax: file.readlines() # reads all lines as a list readline() # reads the next line as a string file.read() # reads the entire file content as a string Example: with open('data.txt', 'r') as file: lines = file.readlines() next_line = file.readline() content = file.read()
File writing methods	Different write methods to write content to a file.	Syntax: file.write(content) # writes a string to the file file.writelines(lines) # writes a list of strings to the file Example: lines = ["Hello", "world"] with open('output.txt', 'w') as file: file.writelines(lines)
Iterating over lines	Iterates through each line in the file using a 'loop'.	Syntax: for line in file: # Code to process each line Example: with open('data.txt', 'r') as file: for line in file: print(line)
Open() and close()	Opens a file, performs operations, and explicitly closes the file using the close() method.	Syntax: file = open(filename, mode) # Code that uses the file file.close() Example: file = open('data.txt', 'r') content = file.read() file.close()
with open()	Opens a file using a with block, ensuring automatic file closure after usage.	Syntax: with open(filename, mode) as file: # Code that uses the file Example: with open('data.txt', 'r') as file: content = file.read()

Pandas		
Package/Method	Description	Syntax and Code Example
read_csv()	Reads data from a '.CSV' file and creates a DataFrame.	Syntax: dataframe_name = pd.read_csv('filename.csv') Example: df = pd.read_csv('data.csv')
read_excel()	Reads data from an Excel file and creates a DataFrame.	Syntax: dataframe_name = pd.read_excel('filename.xlsx') Example: df = pd.read_excel('data.xlsx')
to_csv()	Writes DataFrame to a CSV file.	Syntax: dataframe_name.to_csv('output.csv', index=False) Example: df.to_csv('output.csv', index=False)
Access Columns	Accesses a specific column using [] in the DataFrame.	Syntax: dataframe_name['column_name'] # Accesses single column dataframe_name[['column1', 'column2']] # Accesses multiple columns Example: df['age'] df[['name', 'age']]
describe()	Generates statistics summary of numeric columns in the DataFrame.	Syntax: dataframe_name.describe() Example: df.describe()
drop()	Removes specified rows or columns from the DataFrame. axis=1 indicates columns. axis=0 indicates rows.	Syntax: dataframe_name.drop(['column1', 'column2'], axis=1, inplace=True) dataframe_name.drop(index=[row1, row2], axis=0, inplace=True) Example: df.drop(['age', 'salary'], axis=1, inplace=True) # Will drop columns df.drop(index=[5, 10], axis=0, inplace=True) # Will drop rows
dropna()	Removes rows with missing NaN values from the DataFrame. axis=0 indicates rows.	Syntax: dataframe_name.dropna(axis=0, inplace=True) Example: df.dropna(axis=0, inplace=True)
duplicate()	Duplicate or repetitive values or records within a data set.	Syntax: dataframe_name.duplicated() Example: duplicate_rows = df[df.duplicated()]
Filter Rows	Creates a new DataFrame with rows that meet specified conditions.	Syntax: filtered_df = dataframe_name[(Conditional_statements)] Example: filtered_df = df[(df['age'] > 30) & (df['salary'] < 50000)]
groupby()	Splits a DataFrame into groups based on specified criteria, enabling subsequent aggregation, transformation, or analysis within each group.	Syntax: grouped = dataframe_name.groupby(by, axis=0, level=None, as_index=True, sort=True, group_keys=True, observed=False, dropna=True) Example: grouped = df.groupby(['category', 'region']).agg({'sales': 'sum'})
head()	Displays the first n rows of the DataFrame.	Syntax: dataframe_name.head(n) Example: df.head(5)
Import pandas	Imports the Pandas library with the alias pd.	Syntax: import pandas as pd Example: import pandas as pd

info()	Provides information about the DataFrame, including data types and memory usage.	Syntax: <code>dataframe_name.info()</code> Example: <code>df.info()</code>
merge()	Merges two DataFrames based on multiple common columns.	Syntax: <code>merged_df = pd.merge(df1, df2, on=["column1", "column2"])</code> Example: <code>merged_df = pd.merge(sales, products, on=["product_id", "category_id"])</code>
print DataFrame	Displays the content of the Dataframe.	Syntax: <code>print(df)</code> # or just type <code>df</code> Example: <code>print(df)</code> <code>df</code>
replace()	Replaces specific values in a column with new values.	Syntax: <code>dataframe_name["column_name"].replace(old_value, new_value, inplace=True)</code> Example: <code>df["status"].replace("In Progress", "Active", inplace=True)</code>
tail()	Displays the last n rows of the DataFrame.	Syntax: <code>dataframe_name.tail(n)</code> Example: <code>df.tail(5)</code>

Numpy

Package/Method	Description	Syntax and Code Example
Importing Numpy	Imports the Numpy library.	Syntax: <code>import numpy as np</code> Example: <code>import numpy as np</code>
np.array()	Creates a one or multi-dimensional array.	Syntax: <code>array_1d = np.array([[11111 values]] * 10 #array</code> <code>array_2d = np.array([[11111 values], [11111 values]]) * 20 #array</code> Example: <code>array_1d = np.array([1, 2, 3]) * 10 #array</code> <code>array_2d = np.array([[1, 2], [3, 4]]) * 20 #array</code>
Numpy Array Attributes	<ul style="list-style-type: none">- Calculates the mean of array elements- Calculates the sum of array elements- Finds the minimum value in the array- Finds the maximum value in the array- Computes dot product of two arrays	Example: <code>np.mean(array)</code> <code>np.sum(array)</code> <code>np.min(array)</code> <code>np.max(array)</code> <code>np.dot(array_1, array_2)</code>

